What is CSRF?

CSRF (Cross-Site Request Forgery) is a vulnerability where an attacker tricks a victim's browser into sending a state-changing request to a site the victim is logged into. The action runs with the victim's session without their intent: changing a password, transferring funds, or updating an email. It works because browsers automatically attach cookies. The standard fix is an anti-CSRF token the attacker cannot guess, reinforced by SameSite cookies.

HOW IT WORKS

01 How it works and example

The attacker hosts a page that auto-submits a request to the target:

- A hidden auto-submitting form posts to `https://bank.example/transfer` with the attacker's account as the recipient.
- An `<img src="https://app.example/account/delete">` fires a GET-based state change.
- When the authenticated victim loads the attacker's page, the request runs as them.

CSRF needs the action to rely on cookies alone and to lack an unpredictable token. It does not let the attacker read the response, only cause the action. Examples shown for defensive context.

> TOKENS PLUS SAMESITE
>
> *The reliable defence is an anti-CSRF token: a per-session, unpredictable value required on every state-changing request, which a cross-site attacker cannot supply. SameSite=Lax/Strict cookies add a strong second layer.*

HOW TO DEFEND

- Use anti-CSRF tokens on every state-changing request (synchronizer token or double-submit), validated server-side.
- Set `SameSite=Lax` or `Strict` on session cookies so they are not sent on cross-site requests.
- Require re-authentication or a token for sensitive actions.
- Do not make GET requests state-changing.
- Check Origin/Referer as a supporting control, and test forms for missing token validation.

SOURCES

[1] OWASP: CSRF Prevention

[2] MITRE CWE-352: Cross-Site Request Forgery

[3] OWASP Top 10

Test your application for CSRF and 30+ other classes.

securelayer7.net/learn/application-security/csrf

Open online